CS 7545: Machine Learning Theory	**Fall 2021**
Lecture 12	
Name: Tyler LaBonte	**Professor:** Santosh S. Vempala

1 Weighted Majority

Recall that the perceptron algorithm has mistake bound

$$M \leq \frac{\|\boldsymbol{w}^\star\|_2^2 \|\boldsymbol{x}^\star\|_2^2}{\gamma^2} \propto \frac{1}{\gamma^2}. \tag{1}$$

for learning halfspaces. In a different setting, suppose we are given the predictions of n experts (*e.g.*, data sources) which we can use to make our decision. This information can still be encoded as a point in Euclidean space similarly to the perceptron setting, and we will indeed see a connection to halfspaces at the end of the lecture. The expert setting is more general, though, because we could imagine the experts as paths in a graph (exponential) or probability distributions (infinite). Additionally, we can use bandit feedback or full feedback.

A reasonable goal is to do as well as the best expert. We first try majority vote, *i.e.*, we predict 1 if at least half the experts predict 1, and we predict 0 otherwise. Say we make M mistakes and there is a perfect expert, and our learning strategy is to throw out the majority if they are wrong. Then $M \leq \log n$ since we can make at most $\log n$ mistakes before only the perfect expert remains.

This is obviously unrealistic, so let's say the best expert makes m mistakes and that we restart after we throw out all the experts. Then, for every $\log n$ mistakes of the algorithm, each expert makes at least 1 mistake. So, $M \leq m \log n$. This is still a problem if n is large.

We will employ the weighted majority (multiplicative weights) algorithm to do better:

1. Initialize $w_i \geq 0$ at $w_i = 1$ for all experts i.
2. Predict according to the weighted majority.
3. On a mistake, for each expert i who got it wrong, update $w_i \leftarrow w_i/2$.

We can see that $W = \sum_i w_i$ starts at 1, and on a mistake, it goes down to $3/4$ of its current size (because we halve the majority which is at most half the total weight). So after M mistakes, $W \leq n \left(\frac{3}{4}\right)^M$. And since the best expert makes at most m mistakes, $W \geq \left(\frac{1}{2}\right)^m$. So,

$$\left(\frac{1}{2}\right)^m \leq n \left(\frac{3}{4}\right)^M. \tag{2}$$

Taking logs,

$$-m \leq \log n - M \log \frac{4}{3} \tag{3}$$

$$\implies M \leq \frac{m + \log n}{\log 4/3} \tag{4}$$

$$\approx \frac{3}{2}(m + \log n) \tag{5}$$

We can do even better by randomizing. Suppose we take a parameter ϵ, predict according to expert i with probability w_i/W, and for each expert that makes an error, update $w_i \leftarrow (1-\epsilon)w_i$. Notice that this update happens for each expert that gets it wrong every round, not just when our algorithm makes a mistake.

Define f_t to be the weighted fraction of experts that got it wrong at time t. Then $\mathbb{E}[M] = \sum_{t=1}^{T} f_t$. The total weight at time T is

$$W = n \prod_{t=1}^{T} (1 - \epsilon f_t) \tag{6}$$

and the weight of the best expert i is

$$w_i \geq (1-\epsilon)^m. \tag{7}$$

Taking logs and using the fact that $1 + x \leq e^x$,

$$m(1-\epsilon) \leq \ln n + \sum_{t=1}^{T} (1 - \epsilon f_t) \tag{8}$$

$$\leq \ln n - \epsilon \sum_{t=1}^{T} f_t \tag{9}$$

$$= \ln n - \epsilon \mathbb{E}[M]. \tag{10}$$

So,

$$\mathbb{E}[M] \leq (1+\epsilon)m + \frac{1}{\epsilon} \ln n. \tag{11}$$

We can differentiate to find the optimal ϵ is $\sqrt{\frac{\ln n}{m}}$. Thus,

$$\mathbb{E}[M] \leq m + 2\sqrt{m \ln n}. \tag{12}$$

We can analyze the average expected error to further understand this quantity. Using $m \leq T$,

$$\frac{\mathbb{E}[M]}{T} \leq \frac{m}{T} + \frac{2\sqrt{m \ln n}}{T} \leq \frac{m}{T} + 2\sqrt{\frac{\ln n}{T}}. \tag{13}$$

The second term goes to 0 as $T \to \infty$, so we converge to the best expert.

2 Winnow Algorithm

We can extend the weighted majority idea to our supervised binary classification setting with the winnow algorithm. Suppose $\boldsymbol{x} \in \{0,1\}^n$ and $y \in \{0,1\}$. Then,

1. Initialize $w_i \geq 0$ at $w_i = 1$ for all i.
2. Predict 1 if $\boldsymbol{w}^\top \boldsymbol{x} \geq n$ and 0 otherwise.
3. On a mistake:
   a) If the true label is 1, then for all i with $x_i = 1$, update $w_i \leftarrow 2w_i$.
   b) If the true label is 0, then for all i with $x_i = 1$, update $w_i \leftarrow w_i/2$.

Note that we do not update the w_i for which $x_i = 0$.

Let's start with an easy example. Suppose we apply the winnow algorithm to a disjunction (OR) or r out of n variables, called the relevant variables. Note that the weight of the relevant variables never goes down and is at most n (since then we will always predict correctly). So, the number of positive mistakes $M_+ \leq r \log n$, since at that point we will always predict correctly.

On every positive mistake, the total weight increases by at most n, and on every negative mistake, the total weight decreases by at least $n/2$. Since the total weight must remain positive we have $M_- \leq 2M_+$. So $M = M_- + M_+ \leq 3r \log n$.

Now, suppose we apply the winnow algorithm to a k out of r majority function, where there are r relevant variables and we need k of them for the label to be 1. If $k = 1$ is it a disjunction, if $k = r$ it is a conjunction, and if $k = r/2$ it is a simply majority. We will also add a small change in our update rule: replace 2 by $1 + \epsilon$ for a more general bound.

On a positive mistake, at least k variables get a $1 + \epsilon$ increase, and on a negative mistake, at most $k - 1$ variables get a $1 - \epsilon$ decrease. Since the weight of the relevant variables cannot exceed n, we obtain

$$kM_+ - (k-1)M_- \leq r \log_{1+\epsilon} n. \tag{14}$$

And since the weight must be positive,

$$n + \epsilon n M_+ \geq \frac{\epsilon n}{1+\epsilon} M_-. \tag{15}$$

Rearranging,

$$M_- \leq (1+\epsilon)M_+ + \frac{1+\epsilon}{\epsilon}. \tag{16}$$

Plugging into the first equation,

$$(k - (1+\epsilon)(k-1))M_+ \leq r \log n + (k-1)\frac{1+\epsilon}{\epsilon}. \tag{17}$$

We can again differentiate to find the optimal $\epsilon = \frac{1}{2(k-1)}$. So, $M = \mathcal{O}(kr \log n)$.

3 Winnow for Halfspaces

Previously we had the conditions $\sum_{i=1}^{r} x_i \geq 1$ and $\sum_{i=1}^{r} x_i \geq k$. It is actually simple to extend the winnow algorithm to general halfspaces $\sum_{i=1}^{n} w_i^\star x_i \geq w_0^\star$. We can assume $w_i^\star \geq 0$ by negating the variable if necessary, and we can assume $w_i^\star \in \mathbb{Z}$ by scaling (as long as they are rational to begin with). Furthermore, we can duplicate each variable $W = \sum_i w_i^\star$ times. Now, we have a "learn $w_0^\star$ out of W" problem – we just solved this!

Plugging into the bound we just found,

$$M = \mathcal{O}(w_0^\star W \log(nW)) = \mathcal{O}(W^2 \log(nW)). \tag{18}$$

More generally,

$$M = \mathcal{O}\left(\frac{\|\boldsymbol{w}^\star\|_1^2 \|\boldsymbol{x}\|_\infty^2 \log(n\|\boldsymbol{w}^\star\|_1)}{\gamma^2}\right). \tag{19}$$

Interestingly, both winnow and perceptron have their own advantages. If $\boldsymbol{w}^\star$ is sparse, say $(1, 1, \ldots, 1, 0, 0, \ldots, 0) \in \mathbb{R}^n$, then winnow gives $M = \mathcal{O}\left(\frac{k^2 \log n}{\gamma^2}\right)$ and perceptron gives $M = \mathcal{O}\left(\frac{kn}{\gamma^2}\right)$. But, if $\boldsymbol{w}^\star$ is dense, say $(1/\sqrt{n}, 1/\sqrt{n}, \ldots, 1/\sqrt{n})$, then winnow gives $M = \mathcal{O}\left(\frac{n \log n}{\gamma^2}\right)$ while perceptron gives $M = \mathcal{O}\left(\frac{1}{\gamma^2}\right)$.